Journal of Quality Engineering and Production Optimization

Vol. 6, No. 2, Summer & Autumn 2021

http://jqepo.shahed.ac.ir

Research Paper

DOI: 10.22070/JQEPO.2021.5330.1144

A Combined Data Mining Based-Bi Clustering and Order Preserved Sub-Matrices Algorithm for Set Covering Problem

Reza Kamranrad [1*], **Soheil Soltanzadeh** [1], **Ehsan Mardan** [1]

[1] *Department of Industrial Engineering, Faculty of Engineering, Semnan University, Semnan, Iran*

*** Corresponding Author:** *Reza Kamranrad (Email: r.kamranrad@Semnan.ac.ir)*

Abstract *– This study evaluates a Set Covering Problem (SCP), an extension of the demand covering problem, with several potential applications. The original demand covering problem objective includes the selection of proper locations for a number of available facilities to cover the required demand. The SCP tries to minimize location cost satisfying a specified level of coverage. The SCP problems answer many location problems, e.g., the emergency services sector with alternative facilities that will cover the unavailability of the primary facility or recommender systems where it is desired to fulfill the demand by several available choices. We present a biclustering method to construct biclusters from the distance matrix where a bicluster depicts a subset of demand centers covered by a subset of facilities. According to experiments performed in this study, it is concluded that the proposed method provides high-quality solutions compared with an optimal solution attained from GAMS. Also, for larger problem instances, the proposed method provided solutions with higher quality than GAMS software when the computational time is limited to 1 Hour.*

Keywords*– Biclustering, Data mining, Demand covering, OPSM algorithm, Set covering problem.*

I. INTRODUCTION

The demand covering models include an important part in the location problem. The importance is to attribute proper places to servers where customers' needs could be efficiently covered. Usually, the demand is shown in the form of demand centers. A demand center is satisfied if the server's distance is less than a certain distance. Many applications could be identified for these models, including scheduling of the staff, screening experiments for cancer, etc. (Brotcorne et al., 2002).

There are a lot of conditions and limitations in real-world systems. However, mathematical models have been developed through the years and include these conditions. in coverage models, assumptions are related to three subjects. The first subject is facilities and their properties, like the type of facility and its homogeneity with the system environment. The second subject is about considerations in locating the facility. In recent studies, environmental considerations have also become very important. Finally, satisfying demands and their methods is the third subject.

Manuscript Received: 3- July -2021 & Revised:13-October-2021 & Accepted: 1- Nov -2021
ISSN: 2423-3781

As much as possible, the assumptions made in the mathematical models are tried to be close to reality. The difficult part of developing coverage models or any models related to transportation systems is modeling uncertain parameters like demand patterns. These parameters vary from one system to another. Even it can vary in different time periods for a specific system. Many methods have been developed to estimate these parameters and adapt the model to the real world.

Data mining has recently shown massive interest from academic centers and industries. Such interest arises since data collection and storage has become a common act leading to massive databases. Usually, these databases include a lot of essential data in which conventional methods cannot derive the embedded knowledge. Data mining is commonly related to the process of extracting embedded knowledge and patterns out of large databases. There are various techniques to perform mentioned transformation like classification, regression, clustering etc. This study deals with a specific clustering aspect. Clustering consists of categorizing a set of items into clusters with respect to a set of attributes. Clustering will be performed in a way that the objects inside any cluster possess the most similarity compared with objects outside that cluster. Alternatively, bi-clustering desires to categorize a set of objects and related attributes into classes.

In fact, we present the efficacy of the proposed biclustering (with OPSM method) approach for solving set covering problem through comparison with the results obtained from the GAMS, and also the quality of clusters is assessed with some quality indices.

The remainder of this paper is structured as follows. First, the subject literature is addressed, and then coverage and its model are discussed. In the third part, the proposed two-dimensional clustering concept is described. In the fourth section, the OPSM algorithm is described, and in the fifth section, the application of this algorithm to the model, and finally, the computational results with a general conclusion are presented.

II. LITERATURE REVIEW

Demand covering models are divided into two categories: first, covering problems that a server covers all demand. The second category is maximal, covering problems where the biggest parts of demand are covered by a certain number of servers (Daskin, 1995). In recent years, there have been various applications for such problems. Support coverage, gradual covering, competitive covering, and partnership cover are the most complex types of coverage (Yang et al., 2006). The wide range of applications of these problems has led to fast and efficient algorithms to solve coverage problems. In particular, conventional solutions, such as linear programming, is not efficient. As a result, a large number of problem-solving methods, such as Lagrange algorithms, have been applied to other algorithms like genetic algorithms. However, many of them are not efficient for certain problems and cannot be easily adapted to all of the problems (Kochenberger et al., 2002).

The well-known set covering problem, the main type of covering problems, is applied extensively in many sectors such as personnel scheduling, locating emergency facilities, etc. (Caprara et al., 2000; Toregas, 1971; Toregas and ReVelle, 1972). The problem desires to minimize the server point that can cover the demand. The problem is that the specified set I of demand centers, the J set of candidate points for servers, and a cover matrix $A=(a_{ij})$ are trying to minimize the number of server points so that all demands are covered (Farahani et al., 2012). In the set covering problem, all of the demands should be covered. In a large number of studies, a specified number of service providers may exist that are usually determined on the basis of the allocated budget. The purpose of this paper is to locate these server points so that servers can cover the possible demand. Maximal covering location problem is one of the best-known types of maximal covering problems. If demand center i has the weight of w_i, then the main goal is to provide the servers at a point where the sum of the weights is maximized (Farahani et al., 2012).

Another assumption that is added to these problems is to consider a certain distance between the demand center and service provider in such a way that if the server distance with the demand center is exceeded, demand will not be satisfied. However, this could be unrealistic in practice. Karasakal and Karasakal (2004) considered lower and upper bound for this distance. The coverage was assumed to be a function of the space between the service provider and the demand center. Eiselt and Marianov (2009) maintained the underlying model assumptions but changed the coverage method as a gradual cover. Grossman and Wool (1997) investigated the performance of nine heuristics. Most methods are based on the rounding up of the solution obtained by the linear programming method. Kinney et al. (2007) presented a taboo search method by clustering variables. Different types of models mentioned above indicate different types of coverage. However, all of these problems consist of a matrix $A = (a_{ij})$, where a_{ij} is the service level at point i by server j. Many deterministic and heuristic procedures are provided to deal with these problems (Farahani et al., 2012). Most of these methods have been developed to solve a particular type of problem and cannot be used as a general method to solve these problems; because those methods are based on a special feature or characteristic of problems (Farahani et al., 2012). Consequently, they cannot be widely applied in realistic purposes where it is desired to use different notions of coverage with a similar setting and its effect on the final solution.

Data mining or knowledge extraction from data processing is an attractive field in research. Clustering is one of the most practical concepts in mining methods (Shi et al., 2002). Objects within clusters are relative to a class of attributes (or ratios). Objects in a cluster are more similar than objects that belong to alternative clusters (Glover and Kochenberger, 2006). Homogeneity is computed by considering similarity measures (Boutsinas and Papastergiou, 2008). They usually make clustering algorithms from each cluster, including rows or columns of a data matrix *P (R, C)*. Although in data sets consisting of a huge number of rows and columns, columns often work only under a category of columns (or vice versa). In mentioned highly dimensional environment, the density of objects is usually very limited (Buditjahjanto and Miyauchi, 2011). Biclustering faces this problem in a way that combines rows and clusters concurrently. A bicluster is considered as a sub-matrix that is filled with a cluster of rows and a cluster of columns, and the objects are compared to each other at first comparing objects in the rows when comparisons are compared or similar to the rows when compared to rows. The phrase “bicluster" is familiar for gene expression data clustering in DNA analysis (Cheng and Church, 2000). Tables I and II express the literature about Demand covering problems and data mining more specifically.

TABLE I. SUMMARY OF DEMAND COVERING PROBLEM LITERATURE-PART1

Paper	*Coverage*		*Travel distance*				*Facilities capacity*	
	Total	*Partial*	*Deterministic*	*Probabilistic*	*Stochastic*	*Fuzzy*	*Capacitated*	*Incapacitated*
Brotcorne et al. (2002)	*		*					*
Yang et al. (2006)	*		*				*	
Caprara et al. (2000)	*		*				*	
Toregas et al. (1971)	*		*					*
Farahani et al. (2012)		*	*	*			*	
Pirkul & schilling (1991)		*		*			*	
Karasakal & Karasakal (2004)		*		*				*
Eiselt & Marianov (2009)		*	*				*	
Grossman & Wool (1997)	*		*					*
Kinney et al. (2007)	*		*					*

TABLE II. SUMMARY OF DEMAND COVERING PROBLEM LITERATURE-PART2

Paper	*Differentiation of demand centers*		*Demand satisfaction*		*Solution type*		
	Homogenous	*Heterogeneous*	*Integral*	*Splitted*	*Exact*	*Heuristic*	*Metaheuristic*
Brotcorne et al. (2002)	*		*			*	
Yang et al. (2006)	*		*			*	
Caprara et al. (2000)		*	*		*	*	
Toregas et al. (1971)	*		*		*		
Farahani et al. (2012)		*	*	*	*	*	*
Pirkul & schilling (1991)		*		*	*		
Karasakal & Karasakal (2004)		*	*		*		
Eiselt & Marianov (2009)		*	*		*		
Grossman & Wool (1997)	*		*			*	
Kinney et al.(2007)	*		*				*

In this paper, we use two-dimensional clustering to solve demand covering problems that simultaneously cluster the rows and columns of the coverage matrix. The lines and columns clustering depends on the problem model; whether it is fully clustered or partially determined, it will be specified by an individual. Therefore, solutions could be applied to a variety of problems.

III. PROBLEM MODEL DESCRIPTION

In this paper, we focus on some kinds of problems in which the cost of service for each server is equal to 1, and the value of each demand center is equal to 1. These types of problems are used in many aspects. Depending on what has already been said, the model proposed by Toregas et al. (1971) is defined as:

$$\min \sum_{j=1}^{n} x_j \tag{1}$$

S.t

$$\sum_{j=1}^{n} a_{ij} x_j \geq 1 \ , \qquad i = 1,2,\dots,m \tag{2}$$

$$x_j = 0,1 \ , \qquad j = 1,2,\dots,n \tag{3}$$

Where i is the set of demand centers, j is a set of server sets, a_{ij} is a binary parameter, and if the server j is able to service at point i, it equals to 1 otherwise 0. Also x_j is a binary variable and if the server j satisfies the demand center i, it equals to 1 otherwise 0. The purpose of this model is to minimize the number of servers. The first constraint is to ensure that servers should be able to service a demand center to achieve it. The second constraint also specifies the variant type.

A. The two-dimensional clustering concept (Biclustering)

Using DNA microarray technologies (Yang and Wu, 2006), which have been transcribed a huge number of genes and expression surfaces of each *mRNA* face, are reported under trial conditions. In micro-array analysis, the input matrices $|R|$ and $|C|$ contain the real number indicating the level of expression (both absolute and relative to some references) in each r gene (represented by row) in each c of the experiment conditions (provided with the columns). A bicluster is a category of genes (row) showing constant patterns of expression under a set of experimental conditions or columns. It has been demonstrated that the biclustering models are NP-complete (Cheng and Church, 2000; Ben-Dor et al., 2002; Tanay et al., 2002); this demonstration has been done to examine a cluster of minimal clusters of overlaps (or specifically for conflicting status) or search for a "large" cluster. For instance, there are some adjustments that can be used in polynomial time to determine the bicluster with maximum surroundings (maximizing $|R|$+$|C|$). This issue has not been applied in realistic usages. Usually, the proof is based on the reduction in the group problem computation of two parts in the two-part graph (Tanay et al., 2002; Garey, 1979). Therefore, biclustering algorithms are based on heuristic algorithms (Peeters, 2003; Busygin et al., 2008). We have taken two distinct approaches to recognize biclusters. The first approach is a sort of algorithm that applies an objective function to identify the clusters of "good" clusters to calculate biclusters' quality. They start with random initial clusters (Madeira and Oliveira, 2004; Bergmann et al., 2003; Murali and Kasif, 2002) or typically the entire initial data matrix in a large cluster (Cheng and Church, 2000; Yang et al., 2002), and then try to promote the quality of these clusters using their substitution (deletion, addition, or permutation of rows and columns). Applied procedures do not ensure the diagnosis of the most fitted clusters. Despite little time complexity, they are only an approximation answer. The second approach is using algorithms to ensure that all biclusters of candidate clusters are comprehensive to ensure the diagnosis of the best clusters (Tanay et al., 2002; Lazzeroni and Owen, 2002; Fayyad et al., 1996; Boutsinas, 2013; Crémilleux et al., 2009), because, in an input data matrix *P (R, C)*, the bicluster may exist. Finally, several dual approaches are based on the objective function and research space reduction (Prelić et al., 2006; Ayadi et al., 2009).

The efficiency and simplicity in using this method, especially in large-scale problems, motivated the authors to use it for solving the problem. Furthermore, the proposed method is flexible in modeling user preferences using both local and global similarities.

B. OPSM Algorithm

In this section, we define the order preserved submatrices (OPSM) algorithm and the way it is changed to the sub trail extraction problem in detail. Consider the data matrix *P* with dimensions *n* m*, where *R* - Line and *C* are the columns of a column in *P*. The input vector is the name of row *i* and the name of column j. *S* = (R_s, C_s)is a subset of *P*, where R_s is the subset of *n* row and C_s the subset of *m* columns. There is no need for rows and columns to be continuous in the *P* matrix.

Lemma 1. If there exists a permutation of the C_s, it is a k-matrix. The inputs of each row in R_s are exponentially increasing. If the rows of *x1, x2,* and *x2* are increased from *c1* to *c2,* then *c2,* the basic issue is to determine all large OPSMs of data matrix *P*.

In data preprocessing, any line has been arranged decreasingly, and the values are superseded using the initial name. When the values of two entries are at the same row, one of them is placed more quickly. A sequence structure is repeated when the support of a larger sequence of a minimum support threshold is determined. As a result, the problem of OPSM mining could be considered as data mining. A pattern defines a particular sequence of OPSM in which the sequence pattern is formed of pillars of OPSM and writes the support sequence of the OPSM columns. Most of the existing sequencing methods are followed by finding all the sequences that support the lower support threshold. Clearly, searching algorithm efficiency highly affects the minimum support threshold. A large threshold leads to limited research space. On the other hand, a small threshold leads to high calculations, which increases the complexity of the algorithm diminishes. However, this method avoids some of the important biological and statistical aspects of these processes. Deep OPSMs include platforms with relatively higher columns and rows that could not be processed properly using conventional methods (Wang et al., 2002).

In this paper, to deal with the presented problem, a novel detailed algorithm is presented. The initial stage specifies all common sequences of both lines to develop patterns with the desired lengths that support at least two. Next, the database is examined to compute the row for candidate structure where the length is 2 to find all repeated successive structures with length 2. The third stage is constructing the prefix and storage of frequent sequence patterns (with length 2). The fourth step is the prefix tree and placing nodes in the cluster based on inductive principles to obtain repeated sequence patterns which are in length 3. The algorithm has repeatedly been set up to find OPSM that satisfy the minimal support thresholds. The process acts in such a way that if the larger support threshold is not used for the branches, it is possible to achieve the results that involve all deep OPSMs. All common sub-sequences (ACS) is a variable from conventional (ACS). Longest Common Subsequences (LCS), a classical problem, aims to specify LCS out of a set of sequences (usually two ones). Wang suggested a new approach for computing the similarities along with the two sequences (Xue et al., 2015). Unlike the prior approach, this approach determines similarity with respect to the number of normal sequences among the sequences. *CalACs* is a novel approach to calculate the number of ACS among sequences (Gao et al., 2010). We have improved our ability to obtain all of the normal vectors between sequences. We used a prefix tree to store and scan all normal sequences. Unlike traditional methods for solving multi-dimensional problems, frequent use of column connections can be achieved by using frequent prefix trees. Prefix trees are sorted trees using storage or connected arrays, where nodes form a path from the root to the leaf node. The usual nodes correspond to an empty sequence. The typical node stores the column profiles and maintains the index leaf nodes, which supports the cluster (a cluster of a common subsequence).

Suppose a complete set of ACS is obtained (such as *S*). The R_{ij} indicates the labels of the i_{th} row and j_{th} column. The *Ci* is an element, and *k* is the normal length of the common subsequence, indicating that it is organized. We have added *S* sequences, which we added to the prefix tree, and recorded R_{ij} in the leaf nodes (which support sequences). The traditional methods for constructing a prefix tree are described in the following paragraphs. First, we model the prefix tree in the form of pseudo-sequential. The node $(k+1)$ can be added to the end of the path prior to the leaf node if the first k-1 sequence with length $K+1$ sequence of the prefix k is equal to the path K. As the sequence length $K + 1$ is different along the K sequence, the corresponding leaf node is reviewed, and the lines will be re-counted to obtain the support length of the $K + 1$ path. However, if the data category is highly dimensional and dense [36], the prefix tree will be very large, and when new sequences are added, they occupy a wide space. The navigation operations also take a long time. Therefore, the reduction of computational complexity is necessary. According to inductive principles, if a *K-sequence* length is frequent, all subsequences must be repeated; in other words, if the K - sequence length has subsequences with length K-1, the *K sequence* length should not be repeated. Therefore, if the length of K-1 sequence is formed by K-1 basic item, the length K is not a cluster of frequent trees, and the *K sequence* length should not be added to the candidate tree. Conventional subsequences are applied with length 2 to produce the candidate tree 2. The candidate-2 tree also stores the corresponding root row profiles and all leaf nodes in the corresponding support row and the number of the support lines. In addition, we have used a number of Lines to specify this as a frequent cluster. If the threshold cluster satisfies the support, it maintains it; in other words, it is pruning. After all prune operations are carried out, the repeated-2 tree is obtained, which is the first iteration. Figure 1 shows the overview of the algorithm.

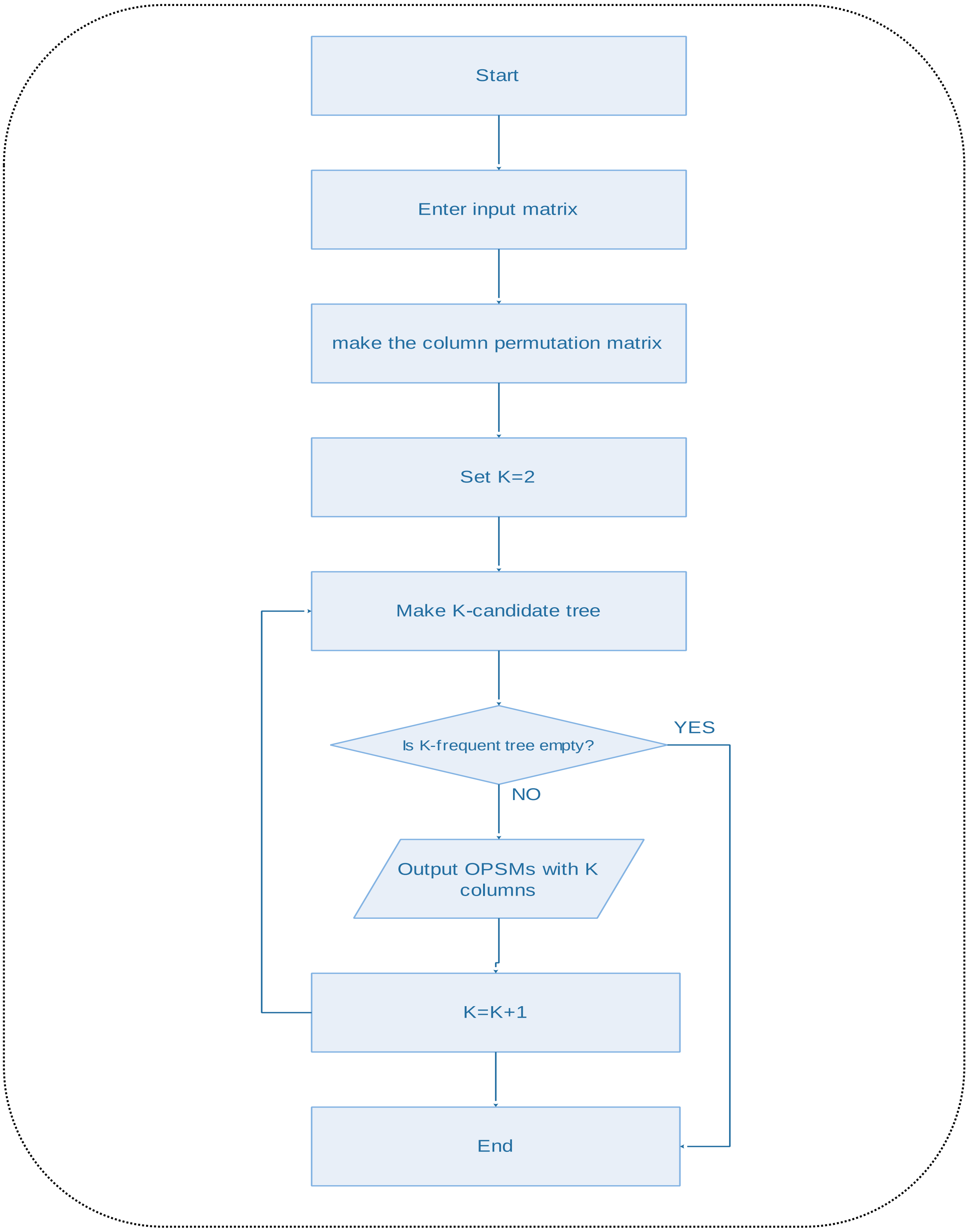
Start
Enter input matrix
make the column permutation matrix
Set K=2
Make K-candidate tree
Is K-frequent tree empty?
YES
NO
Output OPSMs with K columns
K=K+1
End

Fig 1. An overview of the algorithm

C. Applying bicluster to the problem

The presented method first counts candidate sequences. Then executes a selection process. The last selected biclusters cover the entire set. Biclustering method is applied on a binary matrix of input data *P(R, C)* with m rows $R = \{r_1, r_2, \dots, r_m\}$ and n columns $C = \{c_1, c_2, \dots, c_n\}$ where m represents the number of demand centers and n the number of servers. The value of each array (a_{ij}), if equals to 1, means that the server represented by column $c_j, 1 \leq c_j \leq c_n$ covers the demand of point represented by row $r_i, 1 \leq r_i \leq r_m$.

The algorithm runs as follows:

```
$SC(P(R,C))$
{
1 $INPUT\ matrix;$
2 $B = apply\_biclustering(P(R,C));$
3 $Selected\_biclusters = Selected\_columns = Selected\_rows = \emptyset;$
4 $WHILE\ Selected\_rows < |R| \quad DO$
  {
   5 $FOR\ every\ T(R_t, C_t - Selected\_columns), T(R_t, C_t) \in B \quad DO$
  {
   6 $Covered\_rows = \sum \frac{1}{f(r_i)} \quad r_i \in R_t \ \wedge\ \exists c_1, c_2, \dots, c_q \in Selected\_columns \cup C_t\ where\ P(r_i, c_1) = \dots = P(r_i, c_q) = 1;$
     7 $Weight = Weight + \sum Weight(c_i), \quad c_i \in C_t - Selected\_columns;$
     }
     8 $temp(R_s, C_s) = \max(\frac{Covered\_rows}{Weight});$
9 $Selected\_biclusters = Selected\_biclusters \cup temp(R_s, C_s);$
10 $Selected\_columns = Selected\_columns \cup C_s;$
11 $Selected\_rows = Selected\_rows \cup R_s;$
}
12 $RETURN\ Selected\_biclusters;$
}
```

As can be understood from pseudo-code, first, some biclusters are produced through any production algorithm (in which OPSM algorithm is used in this paper). At the beginning of the algorithm, no servers are applied to the demand center. More rows will be covered by columns as we progress through the algorithm. In the next step, the weight of each bicluster will be calculated. Then the ratio of covered rows to weight should be determined. Bicluster with the maximum ratio is the appropriate one. Rows and columns of the bicluster will be added to the current bicluster. At the last step, the optimum bicluster will be displayed.

D. New bicluster columns weight

The frequency function $f(r_i)$, which is the number of arrays with value 1 in each iteration, is updated after the bicluster selection. In each iteration, each line of each vertex must be covered by the minimum number of columns specified as a backup. A bicluster that has the maximum ratio of covered columns to bicluster weight will be added to the final solution. Therefore, in each iteration, rows that are more difficult to cover are selected. Initial biclusters and bicluster production types are effective in the accuracy of our algorithm. The optimal solution is found when the algorithm counts all available biclusters. The OPSM algorithm, the BIMAX algorithm (Crémilleux et al., 2009), and the algorithm which is described in Fayyad et al. (1996), is used to count all biclusters. Figure 2 displays the procedure of the algorithm, along with the classification of various issues.

E. An example of biclustering

The biclustering algorithm proposed here utilizes an enumeration of possible biclusters to ensure the detection of the best biclusters. The proposed algorithm considers binary input data matrix application (In order to be familiar with input matrix specifics, check subsection *C.)*. The proposed algorithm extracts biclusters $S(R_s, C_s)$ where $R_s \subseteq R$, $C_s \subseteq C$ and $\forall i \in R_s, \forall j \in C_s, a_{ij} = 1$.

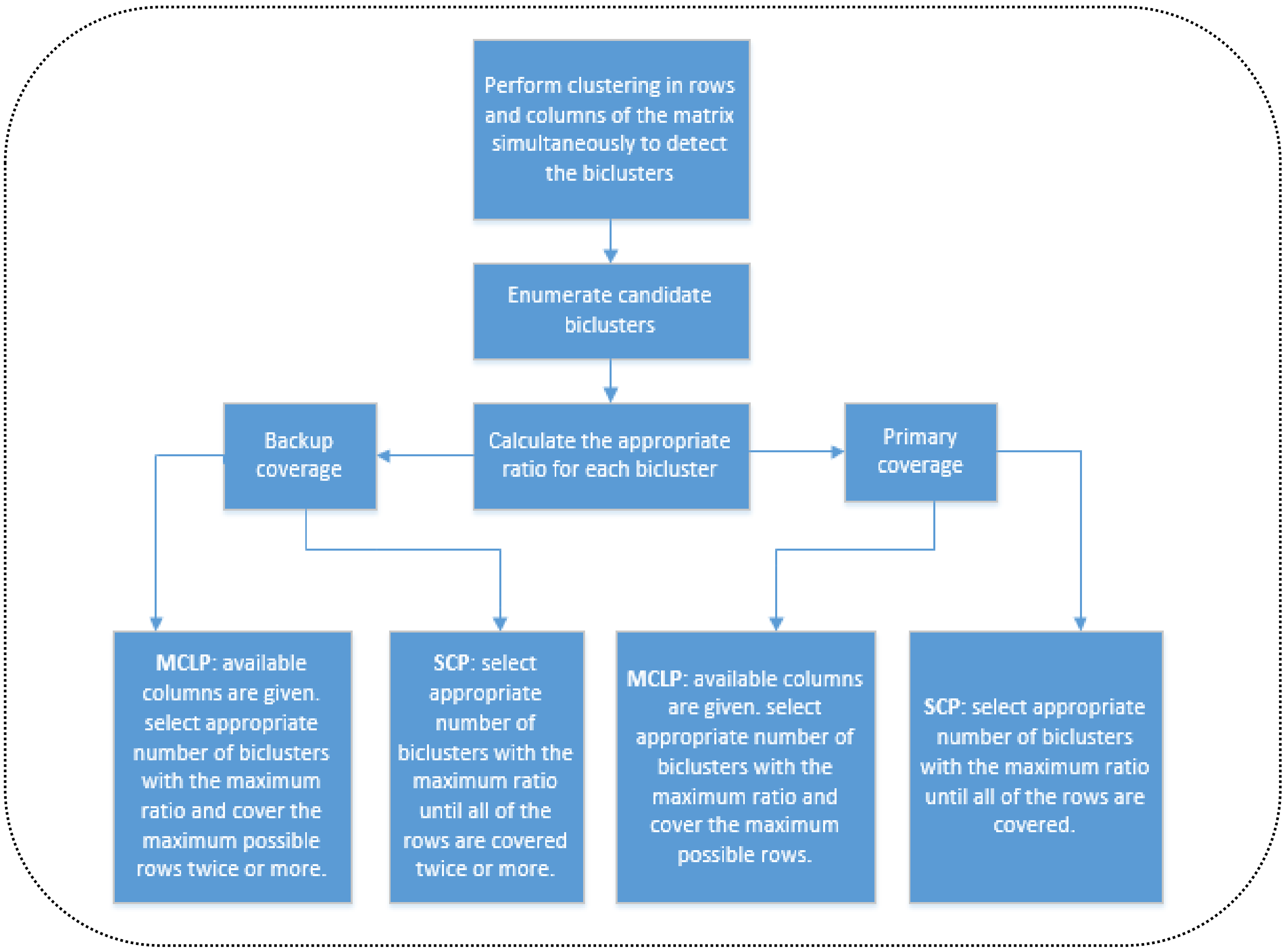
Perform clustering in rows and columns of the matrix simultaneously to detect the biclusters
Enumerate candidate biclusters
Calculate the appropriate ratio for each bicluster
Backup coverage
Primary coverage
MCLP: available columns are given. select appropriate number of biclusters with the maximum ratio and cover the maximum possible rows twice or more.
SCP: select appropriate number of biclusters with the maximum ratio until all of the rows are covered twice or more.
MCLP: available columns are given. select appropriate number of biclusters with the maximum ratio and cover the maximum possible rows.
SCP: select appropriate number of biclusters with the maximum ratio until all of the rows are covered.

Fig. 2 Diagram of the proposed algorithm

The Binary input data matrix on the left of Table III shows that demand center A is satisfied by 1st, 2nd, and 4th servers. In practice, the binary input data matrix is transformed into the matrix $P'(R, C'), C' = \{c'_1, c'_2, \dots, c'_n\}$, depicted in the right of Table III: each demand center is assigned to the column indexes indicating that the servers satisfy it. The proposed algorithm processes the latter matrix by an Association Rule Mining algorithm, like the Apriori algorithm.

TABLE III. – AN EXAMPLE OF BINARY INPUT DATA MATRIX

	1st	*2nd*	*3rd*	*4th*	*5th*
A	1	1	0	1	0
B	1	0	1	0	1
C	0	0	0	1	0
D	0	1	1	0	1

A	1	2	4
B	1	3	5
C	4		
D	2	3	5

Clearly, the attained input data matrix, presented in the right side of the above Table, could be processed as if it were a representation of transactions consisting of itemsets; rows refer to columns itemsets. The application of an Association Rule Mining algorithm, as the Apriori algorithm, led to deriving a set of repetitious itemsets (subsequently applied to attain association rules) with a variety of lengths. It is considered that $fr(s)$ shows a repetitious itemset, where $fr \subseteq I$ and s are its support. For example, $3/5_{50\%}$ is a repetitious itemset attained next to the application of the Apriori algorithm to the matrix shown in the right of Table 1, setting ms = 50% as minimum support. The transactions containing $3/5_{50\%}$ are B and D. Thus, it is obvious that rows B and D, along with the 3rd and 5th columns of the initial input data matrix, form a bicluster, i.e., more than or equal to 50% of the rows (namely B and D) are the most similar to each other (cell values equal to 1) when compared to the 3rd and 5th columns.

IV. COMPUTATIONAL RESULTS

To validate the effectiveness of the proposed methods, we apply it to a range of test examples given in two categories of small-scale examples and classes of examples extracted from the Beasley library. In each example of the first class, both matrices of the covariance matrix have been generated with discrete uniform distribution at a [0, 1] interval. The rest of the examples are originally appeared in references and are directly from the library obtained. Table IV summarizes the characteristics of each category.

Most of the problems are produced in the latter class as low-weight classifiers. Nevertheless, the results show that despite negligible computation time, the cost issues of each unit are balanced (Problem names that have been transformed by transformations in the above matrix) and cannot be effectively solved by careful methods. Following are examples of the OR, based on the difficulty of solving them optimally. For each example, we have solved using GAMS version 24, 1, SCP, and proposed an algorithm. GAMS is considered to be the most advanced package for integrated or linear programming, often used to validate and estimate heuristic algorithms (Beasley, 1996). Note that GAMS does not have the ability to produce an accurate optimal solution for one of the largest examples. As a result, a 3600 second time limit was applied, and the superlative integrated results that were found in this period have entered into the hypotheses. However, by applying the OPSM, the execution time for the cost issues of each unit was between the range of 0.03 and 200 seconds.

As can be seen from Table IV, solutions given by proposed method is near to GAMS solutions. In addition, our method outperformed GAMS in calculation time.

TABLE IV. - SAMPLE ISSUES AND SOME OF THEIR CHARACTERISTICS

Problem	*Number of rows*	*Number of columns*	*Source*	*GAMS Solution*		*OPSM Solutions*		
				Solution Value	*Solution Time*	*Solution Value*	*Solution Time*	*Gap%*
Ins1	15	12	Random	48710.4	33.6	48754.4	6.6	0.09
Ins2	15	16	Random	100832.4	38.4	103494	16.8	2.64
Ins3	20	8	Random	47305.2	37.8	47160.8	3	0.31
Ins4	20	10	Random	96792	20.4	97625.25	27.6	0.86
Ins5	15	8	Random	52014	43.8	52800	7.8	1.51
Ins6	15	10	Random	105816	43.2	106399.2	16.2	0.55
Ins7	20	12	Random	56224.8	63	58621.2	9.6	4.26
Ins8	20	16	Random	114529.2	63	122616	30	7.06
Ins9	15	10	Random	41000.4	33	41064.8	8.4	0.16
Ins10	15	12	Random	85003.2	37.8	85630.4	21.6	0.74
Ins11	50	50	OR Library	60910.8	240.6	63280.8	18	3.89
Ins12	50	100	OR Library	122809.2	265.2	123106.8	138.6	0.24
Ins13	50	100	OR Library	53973.6	109.8	54945.2	28.8	1.80
Ins14	50	200	OR Library	109758	118.8	114583.2	114	4.40
Ins15	50	200	OR Library	59804.4	135.6	60276	31.8	0.79
Ins16	100	50	OR Library	121203.6	112.8	122797.6	43.2	1.32
Ins17	100	50	OR Library	57163.2	128.4	57365.6	3.6	0.35
Ins18	100	50	OR Library	115638	139.8	116245.2	14.4	0.53
Ins19	100	100	OR Library	44236.8	446.4	44561.6	43.8	0.73
Ins20	100	200	OR Library	89734.8	253.8	89563.2	28.8	0.19
Ins21	200	50	OR Library	55556.4	658.8	55188.4	6	0.66
Ins22	200	50	OR Library	112683.6	870	112821.2	100.8	0.12
Ins23	200	50	OR Library	63698.4	1516.2	63996.4	47.4	0.47
Ins24	200	100	OR Library	128235.6	889.8	128174.4	153	0.05
Ins25	200	100	OR Library	57908.4	383.4	57814.4	40.8	0.16
Ins26	200	100	OR Library	117404.4	410.4	117762.4	21	0.30
Ins27	200	100	OR Library	66410.4	738.6	66564.8	42.6	0.23
Ins28	200	100	OR Library	133785.6	1420.2	133528.8	106.2	0.19
Ins29	200	200	OR Library	56911.2	748.8	56097.2	3	1.43
Ins30	200	200	OR Library	115428	763.8	115877.2	67.2	0.39

A. Quality Indices

Another method to assess the clustering quality is the use of indicators for this purpose. These indices are as follows:

1. RMSSTD
2. R Square
3. Partial R Square

Now we describe each one briefly.

B. RMSSTD

This measure is, in fact, the standard deviation of the variables that form a cluster and is derived from the following equation:

$$\text{Compound Variance} = \frac{Sum\ square\ for\ all\ combined\ variables}{Degrees\ of\ freedom\ for\ all} \tag{4}$$

$$RMSSTD = \sqrt{Compound\ Variance} \tag{5}$$

The smaller the value, the more it indicates the homogeneity between the data and the appropriate choice for clustering. If this number allocates some value to itself, it indicates that it is more heterogeneous and better than the done clustering.

C. R-Square

The R Square is the sum of squares between clusters to sum-squares. As the sum of the squares is the sum of squares between clusters and within the clusters, the sum of the least-squares between clusters is smaller, the sum of squares is smaller than the clusters and vice versa. This value varies between 0 and 1, and the bigger it is, indicates the more appropriateness of the clustering.

D. Partial R Square

The difference between the sum of squares of the combination resulting from the creation of a new cluster and the sum of squares between the previous cluster data is called the lost homogeneity. If this number is equal to 0, it is shown that the two clusters are completely homogeneous. The value of Partial R Square is obtained by:

$$SPR = \frac{(Sum\ squre\ from\ clusters - Internal\ sum\ square\ of\ clusters\ before\ combination)}{Total\ sum\ square} \tag{6}$$

The three criteria mentioned above were calculated for the results obtained from the OPSM method and the results are listed in Table V.

TABLE V - VALUES OF MODEL QUALITY INDICES FOR EACH PARAMETER

Problem	*RMSSTD*	*RS*	*SPR*
Ins1	48.97	0.9629	0.0645
Ins2	60.59	0.9239	0.0798
Ins3	47.14	0.9660	0.0620
Ins4	83.90	0.9131	0.1105
Ins5	33.49	0.9769	0.0441
Ins6	62.54	0.9415	0.0823
Ins7	46.56	0.9683	0.0613
Ins8	79.11	0.9170	0.1041
Ins9	50.03	0.9552	0.0658
Ins10	93.42	0.9182	0.0230
Ins11	114.62	0.9351	0.0509
Ins12	239.33	0.8170	0.2152
Ins13	245.59	0.8415	0.2234
Ins14	369.37	0.7042	0.3865
Ins15	340.07	0.7418	0.3479
Ins16	217.87	0.8740	0.1869
Ins17	273.85	0.8397	0.2606
Ins18	278.03	0.8327	0.2662
Ins19	308.57	0.8043	0.3064
Ins20	411.73	0.6957	0.4423
Ins21	364.30	0.7484	0.3798
Ins22	395.30	0.6961	0.4206
Ins23	312.17	0.7912	0.3111
Ins24	480.20	0.6515	0.5324
Ins25	468.15	0.6611	0.5166
Ins26	460.88	0.6680	0.5070
Ins27	497.60	0.5734	0.5554
Ins28	481.58	0.6397	0.5343
Ins29	548.88	0.5649	0.6229
Ins30	559.73	0.5391	0.6372

As can be seen in Table V, in most instances, the value of the *R-square* index is fairly near 1. It shows the appropriateness of the clusters. Furthermore, values of the *Partial R-square* index confirm this.

According to Table V, it can be found that the proposed algorithm has good performance in medium and large samples in large-scale specimens. Obviously, the performance of any algorithm is reduced by increasing the size of the problem. However, the proposed algorithm provides acceptable performance.

V. CONCLUSION

In this study, a clustering-based model for a set covering problem is evaluated. A biclustering method has been proposed which could build relatively high-quality solutions in limited computation times, even for large instances. This was conducted based on an experiment on a series of test instances derived from the OR Library (Beasley, 1996) containing up to 40000 nodes. For most of the problem instances, the results of both quality and computational time were outstanding. In standard problems, the speedup act very well, indicating the advantage of the proposed algorithm compared with the exact method.

For large problems, the results demonstrated that the proposed algorithm leads to better results than GAMS for both CPU time and solution quality. In practice, for the instances with more than 50 and up to 500 columns and rows, OPSM generated solutions with at most a 5% gap and up to 9565 speedups in computation time.

The framework advantages are flexibility, the ability to deal with a great range of coverage problems, and required time. Flexibility arises because any biclustering algorithm with a Boolean input data matrix can substitute the biclustering algorithm within our proposed framework. The results encourage researchers to follow to seek similar techniques for other areas of research. Such techniques might originate new algorithms other than the previously developed algorithms such as Genetic Algorithms, Particle Swarm Optimization, etc. The other advantage of the model is its application in a great range of similar coverage problems. In a practice where it is required to determine plenty of alternatives in limited process time, the simple modification of this method makes it suitable in the application. A modification of the pre-processing procedures could present the extension of the proposed method to accelerate clustering process and thus improve the efficiency of the algorithms.

VI. REFERENCES

Ayadi, W., Elloumi, M. and Hao, J. K. Bio Data mining A biclustering algorithm based on a bicluster enumeration tree: application to dna microarray data. 2009. 2(1): p. 9.

Beasley, J.E. J.J.o.G.O., Obtaining test problems via internet. 1996. 8(4): p. 429-433.

Ben-Dor, A., Chor, B., Karp, R., & Yakhini, Z. Discovering local structure in gene expression data: the order-preserving submatrix problem. 2002. 10(3-4): p. 373-384.

Bergmann, S., Ihmels, J. and Barkai, N. J.P.r.E. Iterative signature algorithm for the analysis of large-scale gene expression data. 2003. 67(3): p. 031902.

Boutsinas, B. J.I.J.o.A.I.T., A new biclustering algorithm based on association rule mining. 2013. 22(03): p. 1350017.

Boutsinas, B. and Papastergiou, T. J.P.R. On clustering tree structured data with categorical nature. 2008. 41(12): p. 3613-3623.

Brotcorne, L., Laporte, G. and Semet, F. Fast heuristics for large scale covering-location Problems, Computers and Operations Research 2002. 29(6): 651-665.

Buditjahjanto, I.A. and Miyauchi, H. J.I.J.o.I.T. D. Making, An intelligent decision support based on a subtractive clustering and fuzzy inference system for multiobjective optimization problem in serious game. 2011. 10(05): p. 793-810.

Busygin, S., Prokopyev, O., & Pardalos, P. M. Biclustering in data mining. Computers & Operations Research. 2008: 35(9), 2964-2987.

Caprara, A., Toth, P. and Fischetti, M. J.A.o.O.R. Algorithms for the set covering problem. 2000. 98(1-4): p. 353-371.

Cheng, Y. and Church G.M. Biclustering of expression data. in Ismb. 2000.

Crémilleux, B., Soulet, A., Kléma, J., Hébert, C., & Gandrillon, O. Discovering knowledge from local patterns in sage data. In Data Mining and Medical Knowledge Management: Cases and Applications. 2009: (pp. 251-267). IGI Global..

Daskin, M.J.N.Y., Network and Discrete Location Wiley. 1995.

Eiselt, H. and Marianov, V. J.S.-E.P.S. Gradual location set covering with service quality. 2009. 43(2): p. 121-130.

Farahani, R. Z., Asgari, N., Heidari, N., Hosseininia, M., & Goh, M. Covering problems in facility location: A review. Computers & Industrial Engineering. 2012: 62(1), 368-407.

Fayyad, U.M., Piatetsky-Shapiro, G. and Smyth., P. Knowledge Discovery and Data Mining: Towards a Unifying Framework. in KDD. 1996.

Gao, B. J., Griffith, O. L., Ester, M., Xiong, H., Zhao, Q., & Jones, S. J. On the deep order-preserving submatrix problem: A best effort approach. IEEE transactions on knowledge and data engineering. 2010: 24(2), 309-325..

Garey, M.R. and Johnson, D. S. J.A.g.t.N.-c., Computer and intractability. 1979.

Glover, F.W. and Kochenberger, G. J.I.J.o.I.T. D. Making, New optimization models for data mining. 2006. 5(04): p. 605-609.

Grossman, T. and Wool, A. J.E.J.o.O.R. Computational experience with approximation algorithms for the set covering problem. 1997. 101(1): p. 81-92.

Hartigan, J.A. J.J.o.t.a.s.a., Direct clustering of a data matrix. 1972. 67(337): p. 123-129.

Karasakal, O. and Karasakal, E.K. J.C.O. Research, A maximal covering location model in the presence of partial coverage. 2004. 31(9): p. 1515-1526.

Kinney, G.W., Barnes, J.W. and Colletti, B.W. J.I.J.o.O.R. A reactive tabu search algorithm with variable clustering for the unicost set covering problem. 2007. 2(2): p. 156-172.

Kochenberger, G., Glover, F., & Alidaee, B. An effective approach for solving the binary assignment problem with side constraints. International Journal of Information Technology & Decision Making. 2002: 1(01), 121-129.

Lazzeroni, L. and Owen, A. J.S.s. Plaid models for gene expression data. 2002: p. 61-86.

Madeira, S.C. and Oliveira, A.L. J.I.A.T.o.C.B. Bioinformatics, Biclustering algorithms for biological data analysis: a survey. 2004. 1(1): p. 24-45.

Murali, T. and Kasif, S. Extracting conserved gene expression motifs from gene expression data, in Biocomputing 2003. 2002, World Scientific. p. 77-88.

Peeters, R. J.D.A.M., The maximum edge biclique problem is NP-complete. 2003. 131(3): p. 651-654.

Pirkul, H. and Schilling, D. A. J.M.S. The maximal covering location problem with capacities on total workload. 1991. 37(2): p. 233-248.

Prelić, A., Bleuler, S., Zimmermann, P., Wille, A., Bühlmann, P., Gruissem, W., ... & Zitzler, E. A systematic comparison and evaluation of biclustering methods for gene expression data. Bioinformatics. 2006: 22(9), 1122-1129..

Shi, Y., Peng, Y., Xu, W., & Tang, X. Data mining via multiple criteria linear programming: applications in credit card portfolio management. International Journal of Information Technology & Decision Making. 2002: 1(01), 131-151.

Tanay, A., Sharan, R. and Shamir, R. J.B. Discovering statistically significant biclusters in gene expression data. 2002. 18(suppl_1): p. S136-S144.

Toregas, C. and ReVelle, C. J.P.i.R.S., Optimal location under time or distance constraints. 1972. 28(1): p. 131-143.

Toregas, C., Swain, R., ReVelle, C., & Bergman, L. The location of emergency service facilities. Operations research. 1971: 19(6), 1363-1373.

Wang, H. and Lin, Z. A novel algorithm for counting all common subsequences. in 2007 IEEE International Conference on Granular Computing (GRC 2007). 2007. IEEE.

Wang, H., Wang, W., Yang, J., & Yu, P. S. Clustering by pattern similarity in large data sets. in Proceedings of the 2002 ACM SIGMOD international conference on Management of data. 2002. ACM.

Xue, Y., Liao, Z., Li, M., Luo, J., Kuang, Q., Hu, X., & Li, T. A new approach for mining order-preserving submatrices based on all common subsequences. Computational and mathematical methods in medicine. 2015. 2015..

Yang, F., Hua, G., Inoue, H., & Shi, J. Two bi-objective optimization models for competitive location problems. International Journal of Information Technology & Decision Making. 2006: 5(03), 531-543.

Yang, J., Wang, W., Wang, H., & Yu, P. /spl delta/-clusters: capturing subspace correlation in a large data set. in Proceedings 18th international conference on data engineering. 2002. IEEE.

Yang, Q. and Wu, X. J.I.J.o.I.T. D. Making, 10 challenging problems in data mining research. 2006. 5(04): p. 597-604.